AD/A-005 692

THE OPTIMAL SELECTION OF SECONDARY INDICES FOR FILES

Mario Schkolnick

Carnegie-Mellon University

Prepared for:

Air Force Office of Scientific Research
Defense Advanced Research Projects Agency

November 1974

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR - TR - 75 - 0196	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD/A005 692
4. TITLE (and Subtitle) THE OPTIMAL SELECTION OF SECONDARY INDICES FOR FILES		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mario Schkolnick		8. CONTRACT OR GRANT NUMBER(s) F44620-73-C-0074
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Dept. Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61101D AO-2466
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE November 1974
		13. NUMBER OF PAGES 16
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Air Force Office of Scientific Research (NM) 1400 Wilson Blvd. Arlington, VA 22209		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

PRICES SUBJECT TO CHANGE

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We consider the problem of finding an optimal set of indices for a file. A general model for a file is assumed together with a probabilistic model of the transactions conducted with it: Queries, Updates, Insertions and Delections. It is shown that all the information assumed for each attribute can be condensed into two parameters and that properties of the optimal solution can be derived from this condensed information. An algorithm to find the optimal set of indices based on these properties is exhibited.

(A)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

THE OPTIMAL SELECTION OF SECONDARY INDICES FOR FILES

Mario Schkolnick
Carnegie-Mellon University
Pittsburgh, Pa. 15213

November 1974

ABSTRACT

We consider the problem of finding an optimal set of indices for a file. A general model for a file is assumed together with a probabilistic model of the transactions conducted with it: Queries, Updates, Insertions and Deletions. It is shown that all the information assumed for each attribute can be condensed into two parameters and that properties of the optimal solution can be derived from this condensed information. An algorithm to find the optimal set of indices based on these properties is exhibited.

This work was supported by Defense Advanced Research Projects Agency under Contract F44620-73-C-0074.

INTRODUCTION

We consider here the problem of selecting a set of attributes for which a secondary index should be provided in order to minimize the expected cost per transaction conducted with a file.

A proper solution to this problem has to consider the system in which the file will exist as well as characteristics such as accessing mechanisms to it and statistical properties of the transactions that are conducted with the file. Some systems which have been implemented use the simple approach of providing indices for all attributes [2] while others do not provide indices at all [6]. For these systems the optimization problem does not exist. Between these two extreme approaches there are systems like XRM [8] which allow the user to specify which domains of the relation should be indexed. Some high level languages have been designed [4] for which the system, while in the process of performing a transaction, creates temporary indices for some attributes. After the transaction is completed, the user is informed of the newly created indices and may then choose to keep them or delete them. Systems such as these and others for which on line transactions may be conducted with a file by users whose demands on the file change in time are best suited for solutions as described here. After collecting statistical properties of the transaction that the current set of users conducts with the file, the system may compute an optimal set of indices to minimize transaction processing time. Then, it creates its optimal index set throwing away those indices which contribute to increase transaction time and creating indices which help to decrease transaction time. Since this overhead processing may be appreciable, the system will probably perform this updating only at some fixed intervals of time.

Recently, a number of studies have appeared in the literature which consider this problem. Some of them have taken an empirical approach [9] while others [7], [10], [14] attempt to formalize the problem to obtain an analytic solution. Since the data base usually exists in a complex environment there are innumerable factors that will influence the index selection problem and, in order to hope for a solution, some assumptions must be made. In [7] the restriction was made that transactions could only specify one attribute value in their specification part while in [14] the transactions were reduced to queries and updates only and the statistical properties considered were minimal. We will now present a model which encompasses a variety of situations by allowing great flexibility in the specification of the statistical properties of the transactions as well as on the storage organization and retrieval mechanism for the indices. The inclusion or exclusion of an attribute in an optimal set of indices will depend on two parameters which are derived for each attribute when the specific properties mentioned above are known to the system. By studying properties of the optimal solution we are able to describe an efficient algorithm which makes use of these pairs of parameters to determine the optimal solution.

SECTION 1

In this section we present the model of the file, the assumptions on storage organization and retrieval, and the types of transactions conducted with it.

We will assume a relational model for a data base [5] and we will consider the problem of index selection when there is a single relation in the data base (the results shown here can be directly extended to a multi-relational data base provided we assume independence between them.

The file F will consist of a set of N vectors (or records) $v = (v_1, v_2, \ldots, v_m)$ where each $v_j \in A_j$, the j-th attribute. Thus $F \subseteq A_1 x A_2 x \ldots x A_m$. We also assume the existence of atoms for the domains [15]. Each attribute will then be a finite set whose elements can appear in a transaction.

We will assume that the use of secondary indices is the only mechanism that will be used to facilitate the search for records in the file. Thus, we will consider here that the file is randomly stored in secondary memory and is not clustered according to any criteria. The approach of clustering records has been studied in the literature [11] but cannot be taken when we assume that the data base is used by several users with different requirements on it. Thus, the time required to retrieve n records will be the time to bring the pages which contain them. Since the records are assumed to be randomly distributed and that selection of an optimal set of indices will result in a small expected number of records to be accessed when processing a transaction, this time will be cn, for some constant c which measures the time required to bring a page to main memory.

Since the method for storing and retrieving indices is a fundamental characteristic of a system using secondary indices for processing transactions we will not impose any restriction on it. The only structure that we impose is that, if an index I_j for the j-th attribute exists, knowledge of a value $a \in A_j$ will produce a list of pointers to records (which we will in the sequel refer to as tid for tuple identifier) in time $f(j,a)$ for some function f. For example, if the indices have the usual two level hierarchical organization and are stored on a disk, $f(j,a)$ could be taken to be $c_1+c_2\beta_j(a)N$ where c_1 is the constant time required to find the head of the appropriate list of tid's, c_2 is the transfer time per tid and $\beta_j(a)$ gives the proportion of the total number N of records that have the value a as its j-th attribute.

Let N be the size of the file. Even though deletions and insertions will be allowed, we assume that the size of the file remains fixed throughout the period between successive computations of optimal sets of indices. For each attribute A_j we will assume a distribution β_j which gives the proportion $\beta_j(a)$ of records in the file having a particular value $a \in A_j$ as its j-th attribute. (One simplifying assumption that sometimes is made is to consider β_j a constant function.)

There will be four types of transactions conducted with the file: Queries, Updates, Insertions and Deletions. It is frequently the case that an Update or a Deletion is specified in two components: a selection component which determines a set of records to be processed, and an action component which, in the case of Updates determines how each record in the set is to be updated, and in the case of a Deletion is null, simply meaning that that set of records which was found is to be removed from the file. As will be discussed below, finding the relevant records implies a number of actions. After they are done, we can no longer assume that any part of an index is in main memory so that the updating on the indices that results due to an Update or a Deletion can be assumed to be independent of the processing of the selection part of the corresponding transaction. For this reason we do not include a selection part in the formal specification of an Update or Deletion.

A Query Q will be specified as $Q = (q_1, q_2, \ldots, q_m)$ where $q_i \in A_i \cup \{x\}$, x is a symbol not in any set A_i. If $q_i \in A_i$, we say that the i-th attribute is specified in the query. Note that a query to our model can result from a true query in the system, or an update or a deletion as discussed above. An actual query is sometimes accompanied by an output part which specifies that some attributes of the set of records found to satisfy the query are to be displayed or further processed. These operations take time independent of the set of attributes for which an index exists and thus can be safely ignored for our analysis.

We will assume, as is done in most systems, that a query is processed as follows:

1. For each specified attribute j, for which an index exists, a list L_j is found containing all tids of records whose j-th attribute have the specified value.
2. From all lists L_j, an intersection list L is formed $L = \cap L_j$. This list contains all tids of records all of whose attributes for which an index exists and which are specified in the query have the specified value. These records will be referred to as partially qualified records.

3. All partially qualified records are brought to main memory where the values of those specified attributes for which there is no index are checked. Those records which are found not to satisfy the query are disregarded (these are sometimes referred to as false drops) and a list of qualifying records (or their tids) is obtained which can be further processed. As was mentioned before any further operation is independent of the chosen index set and will not be considered any further.

Let $\alpha_j(a)$ be the probability that the j-th attribute is specified in a query to have the value $a \in A_j$. For convenience we will denote as p_j the probability that the j-th attribute is specified in a query. Thus,

$$p_j = \sum_{a \in A_j} \alpha_j(a).$$

The expected cost to process step 1 above is given by

$$\sum_{j \in D} \sum_{a \in A_j} \alpha_j(a) f(j,a)$$

where D is the set of attributes for which an index exists. The cost of step 2 can be considered negligible as compared to the cost of 1 and 3. The reason for this is that the intersection list can be constructed in main memory and any processing time spent here is small compared with the cost of interacting with secondary storage. It is not hard to see that the expected number of tids in the resulting intersection list L is given by

$$|L| = N \cdot \prod_{j \in D} [1 - p_j + \sum_{a \in A_j} \alpha_j(a)\, \beta_j(a)]$$

Step 3, as explained at the beginning of this section is proportional to the length of L, $c|L|$. The cost of removing the false drops to form the final list can be considered, as in step 2, to be negligible since it is done in main memory. Thus the expected cost to process a query is given by

$$C_Q = \sum_{j \in D} \sum_{a \in A_j} \alpha_j(a)\, f(j,a) + cN \cdot \prod_{j \in D} [1 - p_j + \sum_{a \in A_j} \alpha_j(a)\, \beta_j(a)],$$

An Update U will be specified as $U = (v{:}u_1, u_2, \ldots, u_m)$ where each $u_i \in A_i \cup \{x\}$ and $v \in F$. The intended meaning for an update is that record v (which can be identified by its tid) has to be updated on those specified attributes A_i (i.e., those for which $u_i \neq x$) to have the new value u_i.

To process an Update, one has to (1) retrieve the record v, update its specified attributes, and store it back again and (2) update all relevant indices. The time required to perform (1) is independent of the existing set of indices and thus we do not consider it. Updating the indices requires reading and writing back the bucket(*) for the old value of the attribute and doing the same for the new value. We will assume that both these operations are performed even if these buckets coincide. (As it turns out, consideration of this fact would only result in a slightly more complicated expression for the parameter K(i) associated to the i-th attribute (see Section 2), but does not otherwise change the nature of the algorithm to find an optimal set of indices.)

Let $f'(j,a)$ be the time required to read and write back the "a" bucket for the j-th index. Then, the expected cost of an update on the j-th index is

$$\sum_{b \in A_j} \mu_j(b) \left[\sum_{a \in A_j} \beta_j(a)\, f'(j,a) + f'(j,b) \right]$$

where the first term inside the square bracket is the cost of updating the old bucket (assuming that the distribution of the number of tids on all buckets of the j-th index is given by β_j) and the second term is the cost of updating the new bucket. Also we assume a probability $\mu_j(b)$ that the value $b \in A_j$ for the j-th attribute will be specified in an Update. Thus, if we have a set D of indices, the expected cost of an Update is given by

$$C_U = \sum_{j \in D} \sum_{a \in A_j} [\gamma_j \beta_j(a) + \mu_j(a)]\, f'(j,a) \quad \text{where}$$

$\gamma_j = \sum_{b \in A_j} \mu_j(b)$ is the probability that the j-th attribute is specified in an update.

* A bucket is the set of all tids of tuples having the same value on an indexed attribute.

Insertions: An insertion is specified as I(v). It requires insertion of the record itself plus the updating of all indices. As was the case for updates the first cost is independent of the index set. Assuming a distribution of values given by β_j, the cost of the second component is

$$\sum_{j \in D} \sum_{a \in A_j} \beta_j(a) f'(j,a).$$

Deletion: A deletion T(v) of the record v requires a similar set of operations as an insertion and the expression for the resulting cost is the same, $C_T = C_I$.

Combining all expressions obtained above, we get that the expected cost per transaction is given by: $E(D) = r_Q C_Q + r_U C_U + r_I C_I + r_T C_T$ where r_Q, r_U, r_I and r_T are respectively, the probabilities that the transaction is a Query, an Update, an Insertion or a Deletion. This expression can also be written as

$$E(D) = \sum_{j \in D} H(j) + G(D) \qquad (1)$$

, where

$$H(j) = \sum_{a \in A_j} \{r_Q \alpha_j(a) f(j,a) + [r_U(\gamma_j \beta_j(a) + \mu_j(a)) + (r_I + r_T)\beta_j(a)] f'(j,a)\}$$

and

$$G(D) = r_Q c N \prod_{j \in D} [1 - p_j + \sum_{a \in A_j} \alpha_j(a)\, \beta_j(a)]$$

The problem of finding an optimal set of indices can be now formally defined as that of finding a smallest set $D \subseteq M = \{1,2,\ldots,m\}$ which minimizes the above expression for E(D).

SECTION 2

Analysis of the Cost Function

A straightforward evaluation of E(D) for all subsets of M would certainly solve the optimization problem. We are interested in finding algorithms which take less than 2^m to obtain the optimal set. Another approach would be to construct chains of sets $D_0 = \phi, D_1, D_2, \ldots, D_k$, with each D_{i+1} a superset of D_i obtained by adding one more element of M, such that $E(D_i)$ resulted in a nonincreasing sequence. Proceeding in this manner, we could find a collection of locally optimal sets D (i.e., all sets D such that, for all j, $E(D) \le E(D \cup \{j\})$ and $E(D) \le E(D - \{j\})$. In general, by following this procedure we may not find the optimal solution among the collection of sets found. For example, assume an optimization problem over a set {A,B,C} whose cost function E'(D) is such that $E'(\phi) = 5$, $E'(\{A\}) = 6$, $E'(\{B\}) = 3$, $E'(\{C\}) = 4$, $E'(\{A,B\}) = 9$, $E'(\{A,C\}) = 7$, $E'(\{B,C\}) = 8$, $E'(\{A,B,C\}) = 2$. The above method would produce the collection {B} and {C} as local optimal solutions reached from ϕ, thus missing the optimal set {A,B,C}.

There is a simple condition on a cost function, which we call the <u>regularity condition</u>, which suffices to guarantee that the above procedure will obtain the optimal solution among the collection of sets which finds. The condition states that, if while performing the procedure, a set D is reached and there is an index $j \notin D$ which increases the cost function, then the index j can be ignored in any subsequent search from D. Formally, we have:

<u>Definition 1</u>: Let E be a cost function defined on subsets of a set S of points. Let $\Delta(D,j) = E(D \cup \{j\}) - E(D)$. Then E is said to be <u>regular</u> if $\Delta(D',j) \ge \Delta(D,j)$ for any point j and sets D, D' for which $D \subseteq D'$ and $j \notin D'$.

Note that if E is regular and for some D and $j \notin D$, $\Delta(D,j) \ge 0$ then, for all $D \subseteq D'$ with $j \notin D'$ we have $\Delta(D',j) \ge 0$.

The following lemma states that, for a regular function, the above procedure succeeds in obtaining an optimal solution.

<u>Lemma 1</u>: Let E be a regular cost function and D a locally optimal set (i.e., $E(D \cup \{k\}) \ge E(D)$ and $E(D - \{k\}) \ge E(D)$). Then $E(D_0) \ge E(D_1) \ge \ldots \ge E(D_n)$ for all sequences of sets $D_0, D_1, \ldots, D_n = D$ satisfying $|D_i| = i$.

<u>Proof</u>: We have to show that for any subset D' of D, $E(D' \cup \{j\}) \le E(D')$ for all $j \in D-D'$ or equivalently, $\Delta(D',j) \le 0$. Assume the contrary, and consider the set

$D'' = D - \{j\}$. Clearly, $D' \subseteq D''$. By assumption, $\Delta(D',j) > 0$, which implies, since E is regular, that $\Delta(D'',j) > 0$. This contradicts the fact that D is a local optimum. ▮

Lemma 1 states that if D is a local optimum, it will be found by the procedure described above because for <u>any</u> chain $\phi = D_0, D_1, \ldots, D_n = D$ with $|D_i| = i$ we have that $E(D_0), E(D_1), \ldots, E(D_n)$ is a nonincreasing sequence. (We note here that an analogous proof shows that Lemma 1 also holds if the sequence of sets is decreasing, i.e., $S = D_0, D_1, \ldots, D_n$ satisfying $|D_i| = |S| - i$, so, in particular, a search which starts from S will also find the global optimum.

There is a class of cost functions which includes our particular cost function, which are regular. They are characterized in the next definition and lemma.

<u>Definition 2</u>: Let K be a function which maps subsets of S to a totally ordered domain with order relation given by $<$. Then K is said to be <u>monotone nonincreasing</u> (mni) if $D \subseteq D'$ implies $K(D') \leq K(D)$.

<u>Lemma 2</u>: Let E be a cost function such that $\Delta(D,j) = E(D \cup \{j\}) - E(D)$ can be written as $\Delta(D,j) = A(j) - B(D,j)$ where, for each fixed j, $B(D,j)$ is mni. Then E is regular.

<u>Proof</u>: Let D', D be subsets with $D \subseteq D'$ and let j be a point $j \notin D'$. We have, by definition, $\Delta(D',j) - \Delta(D,j) = [A(j) - B(D',j)] - [A(j) - B(D,j)] = B(D,j) - B(D',j)$. Since, for fixed j, $B(D,j)$ is mni then $D \subseteq D'$ implies $B(D',j) \leq B(D,j)$. So, $\Delta(D',j) \geq \Delta(D,j)$ as required. ▮

Our first result shows that the cost function we are dealing with is regular.

<u>Theorem 1</u>: Let $E(D) = \sum_{j \in D} H(j) + G(D)$ as defined in (1). Then E is regular.

<u>Proof</u>: By definition, $\Delta(D,j) = H(j) + G(D \cup \{j\}) - G(D) =$

$H(j) - (p_j - \sum_{a \in A_j} \alpha_j(a)\beta_j(a))\, G(D) = H(j) - F(j)G(D)$ where

$F(j) = p_j - \sum_{a \in A_j} \alpha_j(a)\alpha_j(a)$. Since G is clearly mni, E is regular, by Lemma 2. ▮

Since $\sum_{a \in A_k} \alpha_k(a) = p_k$, we have that $0 < F(k) < 1$ (assuming $p_k > 0$ and $|A_k| > 1$.

This assumption is justified since $p_k = 0$ or $|A_k| = 1$ imply $F(k) = 0$ and thus $\Delta(D,k) \geq 0$ so the k-th attribute would not be part of any optimal solution. In the sequel, we will assume this to hold for all attributes).

Other classes of functions satisfying Lemma 2 (and thus the regularity condition) have appeared in the literature. In [3] the following cost function is studied in connection with an optimal allocation of copies of files in an information network with n nodes:

$E(D) = \sum_{k \in D} U_k + \sum_{i=1}^{n} G_i(D)$ where U_k depends only on parameters associated to the k-th node in the network and

$G_i(D) = \lambda_i \min_{k \in D} d_{ik}$ where λ_i is a constant associated with the i-th node and d_{ik} is a cost associated to the link between nodes i and k. E(D) is the cost associated to selecting the set D of nodes as information storage nodes. A result like Lemma 1 but specialized to this function was obtained. Since

$$\Delta(D,j) = U_j + \sum_{i=1}^{n} \lambda_i \left(\min_{k \in D \cup \{j\}} d_{ik} - \min_{k \in D} d_{ik} \right) = U_j - \sum_{i=1}^{n} \lambda_i \left(\min_{k \in D} d_{ik} \dot{-} d_{jk} \right)$$

where $x \dot{-} y =$ <u>if</u> $x \geq y$ <u>then</u> $x-y$ <u>else</u> 0, it follows that this cost function satisfies the conditions of Lemma 2. This implies that it is regular and Lemma 1 holds. Thus, Theorem 1 in [3] is obtained as a special case of Lemma 2.

SECTION 3

Since our cost function is regular we know that a depth first search as described in Section 2 will find the optimal solution. In this section we will show that we do not need to examine all possible nodes which could be reached during an unrestricted search. Thus the time required to find the optimal set will be reduced. This result will be obtained by characterizing properties of the optimal solution.

Definition: Let $A_1, A_2, \ldots, A_m$ be the set of attributes for our file. For each A_j, define a tuple $(F(j),K(j))$ where $F(j)$ is defined as in the proof of Theorem 1 and $K(j) = H(j)/F(j)$ and $H(j)$ has also been defined. Thus we get a set S of m vectors, each having two components. Let $s_i = (F(i),K(i))$, and $s_j = (F(j),K(j))$ be two such vectors. A partial order $\leqslant$ can be defined as follows: $s_i \leqslant s_j$ iff $F(i) \leq F(j)$ and $K(i) \geq K(j)$. If $s_i \leqslant s_j$ then $s_j \geqslant s_i$. A decreasing chain of points in S is a sequence $s_1, s_2, \ldots, s_n$ such that $s_1 \geqq s_2 \geqq s_3 \geqq \ldots \geqq s_n$. The following theorems characterize the set of points in an optimal solution.

Theorem 2: Let S be a set of points as above. If $s_j \in S$ belongs to an optimal solution , then all points $s_i \neq s_j$ such that $s_i \geqslant s_j$ are also contained in an optimal solution.

Proof: Let $s_j \in D'$, an optimal solution. For a given pair s_i, s_j, let $\Delta(D) = \Delta(D,i) - \Delta(D,j)$. (Here, $\Delta(D,i)$ stands for $E(D \cup \{s_i\}) - E(D)$). Assume $s_i \notin D'$. Consider the set $D = D' - \{s_j\}$. Since D' is an optimal solution, $\Delta(D,j) \leq 0$.

Claim: It suffices to show that $\Delta(D,j) < 0 \Rightarrow \Delta(D) < 0$ and $\Delta(D,j) = 0 \Rightarrow \Delta(D) \leq 0$. This follows because, if $\Delta(D,j) < 0$ then $\Delta(D) < 0$ so that $\Delta(D,i) < \Delta(D,j)$. Since we have assumed $s_i \notin D'$, we get $E(D \cup \{s_i\}) < E(D')$, contradicting the optimality of D'. Thus $s_i \in D'$, an optimal set. If, on the other hand, $\Delta(D,j) = 0$ then $\Delta(D) \leq 0$ and so, $\Delta(D,i) \leq 0$. If $\Delta(D,i) < 0$ we get a contradiction as before, and so $s_i \in D'$, an optimal solution. Finally, if $\Delta(D,i) = 0$, $E(D \cup \{s_i\}) = E(D')$ which means $D \cup \{s_i\}$ is also an optimal set, which again, proves the theorem.

To see why the claim holds, $\Delta(D) = F(i)[K(i) - G(D)] - F(j)[K(j) - G(D)] = [F(i) - F(j)][K(j) - G(D)] + F(i)[K(i) - K(j)]$. Since $s_i \geqslant s_j$ we have $F(i) \geq F(j)$ and $K(i) \leq K(j)$ (but $s_j \neq s_i$ so $F(i) > F(j)$ or $K(i) < K(j)$). Thus, if $\Delta(D,j) < 0$ (i.e., $K(j) - G(D) < 0$), then $\Delta(D) < 0$, while if $\Delta(D,j) = 0$, (i.e., $K(j) = G(D)$) then $\Delta(D) < 0$ (if $K(i) \neq K(j)$) or $\Delta(D) = 0$ (if $K(i) = K(j)$). In any case, the theorem is proved. ▌

Theorem 2 says that if an optimal solution contains a point s, then all points in a decreasing chain ending in s are also part of an optimal solution. Using this fact, the search for an optimal solution can be organized as follows: Partition C into the smallest set of disjoint descending chains. Let w be the number of such chains. The set of candidates to be adjoined to a current partial solution D is obtained by considering the subset of independent points among the set of points which are maximal in each chain. Thus, at each step of the search, at most w points have to be considered. If $m_1, m_2, \ldots, m_w$ are the number of points in each chain, the maximal number of sets examined during the entire search will be less than $(1+m_1)(1+m_2) \ldots (1+m_w) \leq (1+m/w)^w$. Assuming that all points have different components, the value of w turns up to be the length of a longest increasing sequence in a permutation of m elements. There is no known expression for the average of this quantity but empirical studies [1] have shown that the asymptotic behavior is $2m^{0.5}$. (Recently, Steckin has shown [13] that this average is bounded above by $em^{0.5}$). So, an upper bound for the average number of sets examined, assuming all permutations being equally likely is $(1+m^{0.5}/2)^{2m^{0.5}} = O(2^{m^{0.5}} \log m)$.

The partial order defined in the definition above, has induced a precedence in the order in which points have to be examined. Theorem 2 established that this precedence was partial as nothing was established for independent points. Theorem 3 will provide conditions under which a precedence can be established for these independent pairs of points. Notice that if a precedence could be established for all independent pairs, then a total precedence would exist and a linear scan would determine the optimal set.

<u>Theorem 3</u>: Let i, j be two independent points in S such that $F(i) < F(j)$ and $K(i) < K(j)$. If $\Delta(D,i) < \Delta(D,j)$ for some D then $\Delta(D',i) < \Delta(D',j)$ for any superset D' of D.

Proof: As in the proof of Theorem 2, we have for $D' \supset D$, $\Delta(D') - \Delta(D) = [F(i) - F(j)][G(D) - G(D')] \leq 0$, since $G(D)$ is mni.

Thus $\Delta(D') \leq \Delta(D) < 0$ as was to be shown. ▮

Theorem 3 says that, if while performing the depth first search procedure, whenever two points in an independent set can be chosen to be included in a set D and the one with smaller value of the F function is preferable to the other (i.e., it decreases more the value of the cost function), it remains preferable at any later stage of the search, which extends the current set D. Thus, at some point during a partial search we may discover a precedence between two points in an independent set. Using these results we may give the following informal description of an algorithm to find the optimal set of indices from a set S specified by tuples $(F(i),K(i))$. The algorithm keeps track of the precedence that exists between points.

1. Initialization ($D_\emptyset$ is the current choice for global optimum, opt is the lowest value of $E(D_\emptyset)$ obtained so far, R is a pushdown stack whose entries are pairs D_i,P_i, where $D_i \subseteq S$, P_i is a directed graph with at most $|S|$ points): $D_\emptyset \leftarrow \phi$, opt $= \infty$. Define an initial directed graph P_{init} as follows: Nodes are all points $i \in S$ such that $\Delta(\phi,i) < 0$. (Points i with $\Delta(\phi,i) \geq 0$ are never included in an optimal solution so they need not be considered.) Node i is directed to j if either $F(i) \geq F(j)$ and $K(i) \leq K(j)$ (thus $i \gg j$ and Theorem 2 applies) or $F(i) < F(j)$, $K(i) < K(j)$ and $\Delta(\phi,i) < \Delta(\phi,j)$ (by Theorem 3). Let $R \leftarrow (\phi,P_{init})$.

```
2. While  $R \neq$ empty do
  begin  $(D,P) \leftarrow R$; (pop the stack)
     Delete all nodes $i$ in $P$ such that $\Delta(D,i) > 0$ or $i \in D$;
     If $P = \phi$ and $\forall j \in D\ (\Delta(D - \{j\},j) < 0)$  (i.e., found local optimal)
     then  if opt > $E(D)$
           then (local optimal found is best so far) begin opt $\leftarrow E(D)$; $D_0 \leftarrow D$ end
     else  begin (let Source (P) be the set of all
                nodes in P having no ingoing directed
                edges.  Note that Source (P) is an independent
                set.  Let P' be the graph obtained by augmenting
                P by joining $i \in$ Source (P) to $j \in$ Source (P)
                whenever $K(i) < K(j)$ and $\Delta(D,i) < \Delta(D,j)$).
                  For each $i \in$ Source (P'), let $R \leftarrow (D \cup \{i\},P')$; (push the stack)
           end;
  end;
```

As was mentioned above, an upper bound on the asymptotic average running time of a deterministic version of this algorithm is $O(2^{m^{0.5} \log m})$ which is a big improvement over 2^m obtained by enumeration. Empirical studies with it have shown that even this reduced upper bound is still much higher than the actual number of nodes visited.

CONCLUSIONS

The problem of index optimization has been solved under very general assumptions and properties of the optimal solution have been found which allows the existence of an efficient algorithm to determine the solution. It is easy to see that previously reported methods for solving this problem ([7], [12], [14]) are special cases of the results shown here.

References

[1] Baer, R. M. and P. Brock, "Natural Sorting over Permutation Spaces," Math. Comp. 22, 1968, pp. 385-410.

[2] Blier, R. and A. Vorkaus, "File Organization in the SDC Time Shared Data Management (TDMS) System," Proceedings of the 1968 IFIP Congress.

[3] Casey, R. G., "Allocation of Copies of a File in an Information Network," Proceedings of the Spring Joint Computer Conference, 1972, pp. 617-625.

[4] Chamberlin, D. D. and R. F. Boyce, "SEQUEL, A Structured English Query Language," presented at SIGFIDET Workshop, Ann Arbor, Michigan, May 1974.

[5] Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," CACM, Vol. 13, No. 6, June 1970.

[6] IBM Information Management System, "IMS/360 Specifications," IBM Manual GH20-4106.

[7] King, W. F., "On the Selection of Indices for a File," IBM Research, RJ 1341, San Jose, January 1974.

[8] Lorie, R. A., "XRM - An Extended (n-ary) Relational Memory," IBM Cambridge Scientific Center, G320-2096, January 1974.

[9] Lum, V. Y. and H. Long, "An Optimization Problem on the Selection of Secondary Keys," ACM Proceedings, National Annual Conference, 1971.

[10] Palermo, F., "A Quantitative Approach to the Selection of Secondary Indexes," IBM Research, RJ730, San Jose, July 1970.

[11] Rothnie, J. B. and T. Lozano, "Attribute Based File Organization in a Paged Memory Environment," CACM, Vol. 17, No. 2, February 1974.

[12] Schkolnick, U, "Optimizing Partial Inversions for Files," IBM Research Report, San Jose, 1974.

[13] Steckin, B. S., "Monotone Subsequences in a Permutation of n Natural Numbers," English Translation in Math. Notes 13, 1973, pp. 310-313.

[14] Stonebraker, M., "The Choice of Partial Inversions and Combined Indices," to appear in Journal of Computer and Information Sciences.

[15] Wong, E. and T. Chiang, "Canonical Structure in Attribute Based File Organization," CACM, Vol. 14, No. 9, September 1971.